(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開平10－289114

(43)公開日　平成10年(1998)10月27日

(51)Int.Cl.⁶	識別記号	FI		
G06F 9/46	340	G06F 9/46	340	B
			340	E

審査請求　未請求　請求項の数4　OL　（全14頁）

(21)出願番号　特願平9-96263

(22)出願日　平成9年(1997)4月14日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川6丁目7番35号

(72)発明者　鈴木　定佳
東京都品川区北品川6丁目7番35号　ソニー株式会社内

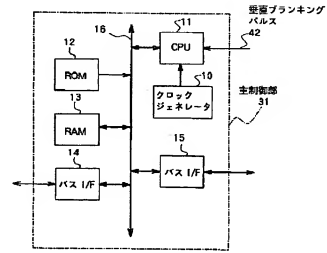
(74)代理人　弁理士　藤島　洋一郎

(54)【発明の名称】マルチタスク制御方法、マルチタスク制御プログラムを記録した記録媒体、およびマルチタスク制御プログラムを組み込んだ電子機器

(57)【要約】

【課題】　所要のメモリサイズを削減できるマルチタスク制御方法、およびマルチタスク制御プログラムを記録した記録媒体、ならびにこのマルチタスク制御プログラムを組み込んだ電子機器を提供する。

【解決手段】　タスクBとタスクCとが共にレディ状態にあって起動要求が競合したときは、原則通り、優先順位に従って、より高い優先順位のタスクに対してCPUの占有権が与えられる。タスクBまたはCの実行中にタスクAから割込みがあったときは、直ちにCPUの占有権をタスクAに与える一方、タスクB（またはC）の実行中にタスクC（またはB）からの割込み要求があったときは、そのタスクの優先順位の如何に係わらず、その実行中のタスクの終了を待ってCPUの占有権を開放する。必要なタスク処理のみがリアルタイム動作し、さほどリアルタイム性を必要としないタスクは、擬似的なリアルタイム動作をする。



1

【特許請求の範囲】

【請求項１】　電子機器の制御上必要な個々の機能に対応して設けられた複数のタスクを制御するための方法であって、

前記複数のタスクのうち、前記電子機器を制御する上で即時実行の要求される特定のタスクに対しては、そのタスクの実行を要求する命令に応じて即時そのタスクを起動させるリアルタイム制御を行う一方、

前記特定のタスク以外のタスクに対しては、そのタスクの実行を要求する命令があった時点において実行状態となっている他のタスクの実行終了を待って起動させるようにしたことを特徴とするマルチタスク制御方法。

【請求項２】　前記特定のタスクの割込みによって実行が中断されたタスクは、その特定のタスクが実行を終了した時点において、優先的に実行を再開することを特徴とする請求項１記載のマルチタスク制御方法。

【請求項３】　電子機器の制御上必要な個々の機能に対応して設けられた複数のタスクを制御するためのマルチタスク制御プログラムであって、

前記複数のタスクのうち、前記電子機器を制御する上で即時実行が要求される特定のタスクに対しては、そのタスクの実行を要求する命令に応じて即時そのタスクを起動させるリアルタイム制御を行う一方、前記特定のタスク以外のタスクに対しては、そのタスクの実行を要求する命令があった時点において実行状態となっている他のタスクの実行終了を待って起動させるようにしたマルチタスク制御プログラムを記録した記録媒体。

【請求項４】　電子機器の制御上必要な個々の機能に対応して設けられた複数のタスクを制御するに際し、前記複数のタスクのうち、前記電子機器を制御する上で即時実行が要求される特定のタスクに対しては、そのタスクの実行を要求する命令に応じて即時そのタスクを起動させるリアルタイム制御を行う一方、前記特定のタスク以外のタスクに対しては、そのタスクの実行を要求する命令があった時点において実行状態となっている他のタスクの実行終了を待って起動させるようにしたマルチタスク制御プログラムを組み込んだ電子機器。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、電子機器の制御上必要な個々の機能に対応したプログラムとして設けられた複数のタスクを制御するためのマルチタスク制御方法、およびマルチタスク制御プログラムを記録した記録媒体、ならびにこのマルチタスク制御プログラムを組み込んだ電子機器に関する。

【０００２】

【従来の技術】従来より、例えば、一般家庭用または業務用のテレビジョン受像機やコンピュータ用ディスプレイ装置等の映像表示装置においては、操作キーから入力されたデータの処理をするタスク、画面表示の制御を担

2

当するタスク、あるいは入力される映像・音声信号における同期信号の周波数測定を担当するタスク等、映像表示装置が有する各機能ごとにタスクをそれぞれ独立に作成し、これらのタスクをＣＰＵがＯＳに従って統括管理するようになっている。このように複数のタスクに機能を分担して割り当てることによって、多人数によるプログラムの並行開発が可能となる等、プログラムの開発効率を向上させることができる。

【０００３】このような複数のタスクを管理するマルチタスク制御型のＯＳにおいては、それぞれのタスク処理をリアルタイムに実行できることが理想であり、このリアルタイム処理を実現するため、従来より各種のリアルタイムＯＳ（ＲＴＯＳ）が提案されている。

【０００４】図１５は、従来のリアルタイムＯＳによるマルチタスク制御方法を表すものである。この例では、３つのタスクＡ～ＣによるＣＰＵの占有状態をＯＳが制御するものとし、このうちタスクＡの優先順位が最も高く（Ｐ＝１）、タスクＢの優先順位が次に高く（Ｐ＝２）、タスクＣの優先順位がもっとも低い（Ｐ＝３）ものとする。

【０００５】図１５に示したように、当初、最下位優先順位のタスクＣが実行状態であったとする。この状態で、時刻τ１においてタスクＢの起動要求があると、ＯＳはタスクＣによるＣＰＵの占有を一旦中断させ、タスクＢにＣＰＵの占有権を渡す。このとき、タスクＣは、図１６に示したように、中断時におけるＣＰＵの状態を格納した各種レジスタの内容をＲＡＭ（Random Access Memory）内に確保したタスクＣ用スタック領域に退避（プッシュ）してその後の処理の再開に備える。さらに、時間が経過し、時刻τ２において、優先順位が最も高いタスクＡの起動要求があると、ＯＳはタスクＢによるＣＰＵの占有を一旦中断させ、タスクＡにＣＰＵの占有権を渡す。このとき、タスクＢは、中断時におけるＣＰＵの状態を格納した各種レジスタの内容をＲＡＭ内のタスクＢ用スタック領域（図１６）に退避（プッシュ）してその後の処理の再開に備える。さらに、時間が経過して、時刻τ３においてタスクＡの実行が終了すると、タスクＡはＣＰＵの占有を開放する。この時点では、タスクＢおよびタスクＣが再開待ち状態になっているが、タスクＢの方がタスクＣよりも優先順位が高いので、ＯＳは、ＣＰＵの占有権をタスクＢに与える。このとき、タスクＢ用スタック領域（図１６）に退避されていた各レジスタデータがＣＰＵ内のレジスタ群にそれぞれ戻され（ポップされ）、中断時のＣＰＵの状態が復元されて、タスクＢの実行が再開される。さらに、時間が経過して、時刻τ４においてタスクＢの実行が終了すると、タスクＢはＣＰＵの占有を開放する。この時点では、タスクＣのみが再開待ち状態になっているので、ＯＳは、ＣＰＵの占有権をタスクＣに与える。このとき、タスクＣ用スタック領域（図１６）に退避されていた各レジス

タデータがＣＰＵ内のレジスタ群にそれぞれ戻され、中断時のＣＰＵの状態が復元されて、タスクＣの実行が再開される。なお、図１６において、Ａ用スタック領域は、タスクＡの実行中に外部からの割込みが生じた場合やサブルーチンコールを行った場合等に、その時点のＣＰＵのレジスタ群の内容を一時退避するためのものである。

【０００６】このように、従来は、各タスクに優先順位を割りつけると共に、あるタスクの実行中に優先順位の高い他のタスクからの割込み要求があったときは、その実行中のタスクの処理を必ず中断してその割込み要求をした上位タスクの実行を即時確保するようにしていた。このため、ＯＳが管理するすべてのタスクについてそれぞれ専用のスタック領域を用意する必要があった。

【０００７】【発明が解決しようとする課題】ところで、このようなＲＴＯＳを用いて構築したシステムとして、ＴＲＯＮ(The Realtime Operating system Nucleus) と呼ばれるコンピュータアーキテクチャ体系が知られているが、その中に、機器組込制御用マイクロコンピュータの標準ＯＳを目指したμＩＴＲＯＮ（micro Industrial TRON ；マイクロアイトロン）と呼ばれるものがある。このμＩＴＲＯＮでは、汎用性を確保すべく、個々の電子機器からすれば不必要な機能についてまでリアルタイム性を追求している。例えば、ＯＳの管理下にあるすべてのタスクに対してリアルタイム動作が要求されていたり、不必要なシステムコールが行われる等、過剰な仕様となっている場合が多い。具体的には、ある電子機器からみれば不要なタスク状態管理用フラグが存在していたり、あるいは不要なプログラムが含まれていること等がある。ここで、通常、各タスクはプログラムとしてＲＯＭ（Read Only Memory）に格納されるが、上記のように不要な部分が多く含まれていると、限られた記憶容量しかもたないＲＯＭというハードウェア資源が無駄に消費されることとなり、このことがプログラム開発上で大きな障害の原因となる場合があった。

【０００８】また、この種のＯＳでは、各タスクに平等にリアルタイム処理を保証する必要があることから、図１５において説明したように、タスクＡ～Ｃのそれぞれについて専用のスタック領域を用意する必要がある。このため、必要なスタックメモリ容量は、タスク数に応じて増加することになる。

【０００９】そこで、必要となるスタックメモリの容量を削減すべく、スタックメモリを共有化することが考えられるが、この場合には、その共有化のためのプログラム等を別途ＯＳに組み込む必要が生じ、却ってＲＯＭの記憶容量を多く消費する結果となる。

【００１０】本発明はかかる問題点に鑑みてなされたもので、その目的は、所要のメモリサイズを削減することができるマルチタスク制御方法、およびマルチタスク制御プログラムを記録した記録媒体、ならびにこのマルチタスク制御プログラムを組み込んだ電子機器を提供することにある。

【００１１】【課題を解決するための手段】本発明に係るマルチタスク制御方法は、複数のタスクのうち、電子機器を制御する上で即時実行の要求される特定のタスクに対しては、そのタスクの実行を要求する命令に応じて即時そのタスクを起動させるリアルタイム制御を行う一方、特定のタスク以外のタスクに対しては、そのタスクの実行を要求する命令があった時点において実行状態となっている他のタスクの実行終了を待って起動させるようにしたものである。ここで、特定のタスクの割込みによって実行が中断されたタスクは、その特定のタスクが実行を終了した時点において、優先的に実行を再開するようにすることができる。

【００１２】本発明に係るマルチタスク制御プログラムを記録した記録媒体は、複数のタスクのうち、電子機器を制御する上で即時実行が要求される特定のタスクに対しては、そのタスクの実行を要求する命令に応じて即時そのタスクを起動させるリアルタイム制御を行う一方、特定のタスク以外のタスクに対しては、そのタスクの実行を要求する命令があった時点において実行状態となっている他のタスクの実行終了を待って起動させるようにしたマルチタスク制御プログラムを記録したものである。

【００１３】本発明に係るマルチタスク制御プログラムを組み込んだ電子機器は、複数のタスクのうち、電子機器を制御する上で即時実行が要求される特定のタスクに対しては、そのタスクの実行を要求する命令に応じて即時そのタスクを起動させるリアルタイム制御を行う一方、特定のタスク以外のタスクに対しては、そのタスクの実行を要求する命令があった時点において実行状態となっている他のタスクの実行終了を待って起動させるようにしたマルチタスク制御プログラムを組み込んだものである。

【００１４】本発明に係るマルチタスク制御方法では、即時実行が要求される特定のタスクに対してのみ限定的にリアルタイム制御が行われ、それ以外のタスクに対してはいわば擬似的なリアルリイム制御が行われる。

【００１５】【発明の実施の形態】以下、本発明の実施の形態について図面を参照して詳細に説明する。

【００１６】図１は本発明の一実施の形態に係るマルチタスク制御方法が適用される電子機器としての映像表示装置の概略構成を表すものである。この映像表示装置は、主制御部３１と、ユーザインタフェイス（Ｉ／Ｆ）制御部３２と、オンスクリーンディスプレイ（ＯＳＤ）制御部３３と、不揮発性メモリ３４と、偏向制御回路３５とを備え、シリアルバス３０によって相互に接続され

ている。本装置はまた、I^2C(Inter Integrated Circuit)バス３６によって主制御部３１に接続されたビデオ制御回路３７と、ビデオ制御回路３７およびＯＳＤ制御部３３に接続されたＲＧＢデコーダ３８と、このＲＧＢデコーダ３８の出力端側に接続されたＣＲＴ（陰極線管）３９とを備えている。

【００１７】主制御部３１は、映像表示装置全体の動作を制御するためのもので、ＣＲＴ３９における表示画面の各垂直帰線期間のタイミングごとに偏向制御回路３５から入力される垂直ブランキングパルス（Ｖ－ＢＬＫ－Ｐｕｌｓｅ）４２に同期してバスアクセスを行い、全体の動作を制御するようになっている。具体的には、ＲＧＢ入力端子からビデオ制御回路３７に入力されたＲＧＢ信号（Ｒ，Ｇ，Ｂの各色信号）４３の状態を制御したり、ユーザインタフェイス制御部３２を介し、図示しない音量キーやメニューキー等のパネル操作キーから入力され、またはリモートコントローラ（図示せず）から受光部を経て送られてきたコマンドデータ等に応じて、シリアルバス３０に接続された各デバイスや図示しない電源供給状態表示用の電源ＬＥＤ（発光ダイオード）等を制御するようになっている。

【００１８】ユーザインタフェイス制御部３２は、パネル操作キーや受光部等からの入力データをサンプリングし、主制御部３１からの問い合わせに応じてそのサンプリングデータをシリアルバス３０を介して送出したり、主制御部３１からの要求に応じて電源ＬＥＤを点灯させる等の制御を行う。ＯＳＤ制御部３３は、主制御部３１からの指示に基づいてＣＲＴ３９に映像と重畳させて表示させるキャラクタを表すカラーデータをＲＧＢ信号４１としてＲＧＢデコーダ３８に出力するようになっている。偏向制御回路３５は、ＣＲＴ３９の水平および垂直ビームの偏向を制御するためのもので、この回路内のいくつかのデバイスがシリアルバス３０を通して主制御部３１から制御されるようになっている。この回路は、例えば水平同期周波数が３１．５ｋＨｚ、垂直同期周波数が７０／６０／５０Ｈｚという走査周波数で偏向制御を行う、いわゆるシングルスキャンシステムを構成している。ビデオ制御回路３７は、I^2Cバス３６を通して主制御部３１から与えられる指示に基づいて、外部ＲＧＢ入力端子から入力されたＲＧＢ入力信号４３に所定の信号処理を施し、ＲＧＢ映像信号４５としてＲＧＢデコーダ３８に入力するという制御を行う。ＲＧＢデコーダ３８は、ビデオ制御回路３７からのＲＧＢ映像信号４５と、ＯＳＤ制御部３３からのＲＧＢ信号４１とを混合してＣＲＴ３９に送出するようになっている。

【００１９】図２は、図１における主制御部３１の概略構成を表すものである。この主制御部３１は、クロックジェネレータ１０から出力される所定の周波数のクロックに同期して動作するＣＰＵ１１と、各種のプログラムを格納したＲＯＭ１２と、ワーク用メモリとしてのＲＡＭ１３と、I^2Cバス３６とのバス接続を行うバスインタフェイス１４と、シリアルバス３０とのバス接続を行うバスインタフェイス１５と、これらを相互に接続する内部バス１６とを備えている。

【００２０】ＣＰＵ１１は、ＡＬＵ（算術論理演算部）、インストラクションデコーダ、累算器（アキュムレータ）、およびタイミング制御部等の各種回路のほか、インストラクションレジスタ、フラグレジスタ、プログラムカウンタ等の専用レジスタ、その他、汎用レジスタ等（いずれも図示せず）を備えて構成されている。このＣＰＵ１１には、図１の偏向制御回路３５から一定周期で垂直ブランキングパルス４２が入力されると共に、その内部では、後述するオペレーティングシステムが各タスクを制御する際の基準となるＯＳ基準割込信号が一定周期で発生するようになっている。

【００２１】ＲＯＭ１２は、オペレーティングシステム（ＯＳ）の本体部（以下、単にＯＳという。）のほか、ＣＰＵが行うべき動作を機能ごとに分担するタスクプログラム（以下、単にタスクという。）等を格納するためのものである。本実施の形態では、説明の簡単のため、図３に示したように、ＯＳとかみ合う形で３つのタスクＡ，Ｂ，Ｃが組み込まれているものとする。後述するように、ＯＳはタスクＡ～Ｃのそれぞれからの要求あるいは命令（システムコール）に応じて、ＣＰＵ１１の占有権を各タスクに与える制御を行う。

【００２２】タスクＡ～Ｃは、それぞれ、属性１または属性２のいずれかを有する。ここで、属性１のタスクは優先順位が最も高いタスクであり、上記した垂直ブランキングパルス４２による割り込みがあった場合のシステムコールに応じて他のタスクの実行を中断させ、自らが即時実行状態となることができるタスクである。そして、この属性１のタスクは、必ずタスクのプログラムの最初から実行されるようになっている。本実施の形態では、タスクＡのみが属性１に属するものとする。ここで、タスクＡは、本発明における特定のタスクに対応し、タスクＢおよびタスクＣは本発明における特定のタスク以外のタスクに対応する。

【００２３】一方、属性２のタスクは、優先順位が２位以下の任意の順位に設定可能なタスクであり、垂直ブランキングパルス４２による割り込みがあった場合に即時実行状態となることはない。そして、この属性２のタスクは、属性１のタスクによって実行を中断された場合を除き、必ず各タスクのプログラムの最初から実行されるようになっている。本実施の形態では、タスクＢおよびタスクＣが属性２に属するものとし、このうち、タスクＢの優先順位が２位、タスクＣの優先順位が３位であるとして説明する。

【００２４】本ＯＳを図１に示した映像表示装置に適用する場合においては、垂直ブランキングパルス４２による割込みによって起動するタスクＡは、例えば内部バス

アクセスを担当するタスクであり、タスクBおよびタスクCは、例えばキースキャンやOSDデータの管理を担当するタスクである。

【0025】RAM13は、CPU11内の上記したレジスタ群の内容を一時的に退避するためのスタック領域や、各種のバッファメモリ領域等のワークエリアとして用いられるもので、例えばDRAM(Dynamic Random Access Memory)によって構成される。

【0026】図4（a）はCPU11のレジスタ群を表し、同図（b）はROM12およびRAM13におけるメモリアドレスの割り当て状態を表すものである。この図の（a）に示したように、CPU11のレジスタ群には、プログラムカウンタ、インストラクションレジスタ、フラグレジスタ、およびその他の汎用レジスタ等が含まれている。また、この図の（b）に示したように、ROM12およびRAM13には、連続したメモリアドレスが割り当てられており、このうち、RAM13には、OS用スタック領域SO と、BC共用スタック領域SBCと、A用スタック領域SA と、他のRAM領域とが割り当てられている。ここで、OS用スタック領域SOは、OSがタスクA～Cのいずれかに CPU11の占有権を開放する際にレジスタ群の内容を一時的に退避させる領域であり、A用スタック領域SAは、タスクAの実行中に外部からのより緊急度の高い割込みが生じた場合や、自らサブルーチンコールを行った場合等に、その時点のCPUのレジスタ群の内容を一時退避するためのものである。また、BC共用スタック領域SBCは、タスクB（またはC）がOSもしくは他のタスクにCPU11の占有権を開放する際にレジスタ群の内容を一時的に退避させる領域である。このBC共用スタック領域SBCは、タスクBとタスクCとにより共用される領域であるが、後述するように、OSは、BC共用スタック領域SBCが2つのタスクB，Cによって同時に使用されることがないように各タスクの状態制御を行う構成になっている。この点が、本発明の特徴の1つをなしている。

【0027】また、図4（b）における他のRAM領域には、タスクA～Cのそれぞれについて、スタックポインタ（SP）が指すスタック領域のメモリアドレスを格納するためのスタックポインタ用バッファ（A用SPバッファ、B用SPバッファ、およびC用SPバッファ）が設けられている。なお、本図では、これらのバッファの図示を省略している。

【0028】図5は各タスクにおける状態遷移を表すものである。この図に示したように、各タスクは、実行（RUN）、レディ（READY）、SPLレディ（SPECIAL READY）、待ち（WAIT）、および休眠（DORMANT）の各状態を取り得るようになっている。ここで、実行状態とはタスクが実行中である場合をいい、休眠状態とは、タスクがまだ起動されていない状態、またはタスクが終了した状態をいう。待ち状態とは、そのタスクを実行するための条件が整うのを待っている状態であり、具体的には、指定された時間の経過を待っている状態と、タスクが指定したセマフォア(Semaphore) が資源を獲得するのを待っている状態とがある。レディ状態とは、自タスク側の実行の準備は整っているが、他のタスクが実行中である状態をいう。SPLレディ状態とは、垂直ブランキングパルス42による属性1のタスク（ここではタスクA）の実行開始によって実行が中断されているタスクの状態をいい、レディ状態の中の特殊な状態である。このSPLレディ状態にあるタスクは、属性1のタスクの実行が終了したときは、設定された優先順位に係わりなく（すなわち、他の優先順位の高いタスクがレディ状態にあったとしても）実行が再開されるタスクである。このSPLレディ状態を設けた点が本発明の特徴の一つをなしている。

【0029】次に、図5および図6を参照して、以上のような構成の主制御部31におけるCPU11の動作を説明する。なお、図6は、タスクA～Cのいずれかのタスク（以下の説明では自タスクと表現する。）がOSの制御の下でどのようにして自己の状態を遷移させるかを表すものである。ここでは、当初のタスク状態が休眠状態となっているものとして説明する。

【0030】図5において、自タスクが休眠状態にある場合において、自タスクを起動させるためのスタート命令が他のタスクからOSになされると（図6ステップS101；Y）、自タスクは直ちにレディ状態に遷移する（ステップS102）。ここで、自タスクが即時実行可能状態であって（ステップS103；Y）、かつ、その時点でレディ状態となっているタスクの中で自タスクの優先順位が最も高く（ステップS104；Y）、しかもその時点でSPLレディ状態にある他のタスクが存在しないときには（ステップS105；Y）、自タスクは、直ちに実行状態へと遷移する（ステップS106）。また、自タスクの優先順位が最も高いが（ステップS104；Y）、その時点でSPLレディ状態にある他のタスクが存在するときには（ステップS105；N）、自タスクは実行状態に遷移せず、そのままレディ状態を維持する。

【0031】一方、ステップS101においてスタート命令があってレディ状態に移った場合でも、自タスクが即時実行可能状態でないときには（ステップS103；N）、自タスクはそのままレディ状態を維持するが、この状態で自タスクを待ち状態にするためのウェイト命令が他のタスクからOSになされると（ステップS107；Y）、自タスクは待ち状態に遷移する（ステップS108）。さらに、この状態で他のタスクから待ち状態解除命令がなされた場合、あるいは、所定時間の経過により自動的に待ち状態解除が行われるような設定になっていた場合にはその設定された時間の経過により、自タスクは再びレディ状態に遷移する（ステップS10

２）。

【００３２】図６のステップＳ１０６において実行状態に遷移したのちは、次のように動作する。すなわち、自タスクを待ち状態にするためのウェイト命令を自タスクからＯＳに対して行うことにより（ステップＳ１１０；Ｙ）、自タスクは実行状態から待ち状態へと遷移する（ステップＳ１０８）。また、属性１のタスクが起動して自タスクの実行が中断されたときは（ステップＳ１１１；Ｙ）、自タスクはＳＰＬレディ状態へと遷移する（ステップＳ１１４）。また、自タスクを休眠状態にするための終了命令を自タスクからＯＳに対して行うことにより（ステップＳ１１２；Ｙ）、自タスクは実行状態から休眠状態へと遷移する（ステップＳ１１３）。それ以外の場合は（ステップＳ１１０；Ｎ，ステップＳ１１１；Ｎ，ステップＳ１１２；Ｎ）、その実行状態を維持する。

【００３３】ステップＳ１１４においてＳＰＬレディ状態に遷移したのちは、その状態遷移の原因となった属性１のタスクの実行状態の終了を待って（ステップＳ１１５；Ｙ）、自タスクは実行状態となり、実行を再開する（ステップＳ１０６）。このように、ＳＰＬレディ状態にある自タスクは、属性１のタスクの実行状態の終了時点において自分よりも優先順位の高い他のレディ状態のタスクが存在していたとしても、それに優先して自らが実行状態となることができる。

【００３４】次に、図７～図１５を参照して、本ＯＳによるタスクＡ～Ｃの制御に係る具体例を説明する。ここで、図７はＯＳおよびタスクＡ～Ｃにおける時刻経過に伴うＣＰＵの占有状況の一例を表し、図８、図１４および図１５は、それぞれ、図７の状況Ｃ１～Ｃ２を詳細に表すものである。また、図９～図１３は、図８の状況Ｃ１におけるＣＰＵ１１および各メモリの状態変化を表すものである。

【００３５】本実施の形態では、タスクＡ（図７（ｄ））は垂直プランキングパルス４２（同図（ｃ））に同期して直ちに起動して実行状態となり、タスクＢ（同図（ｅ））は、ＣＰＵ１１の内部で一定周期で発生するＯＳ基準割込信号２（同図（ａ））のタイミングで起動をかけられるものとする。また、タスクＣ（同図（ｆ））は、ＣＰＵ１１の内部で一定周期で発生する他のＯＳ基準割込信号１（図７（ｂ））の２周期ごとのタイミングで起動をかけられるなるようになっているものとする。

【００３６】まず、図８および図９～図１３を参照して、図７の状況１における動作を詳細に説明する。

【００３７】図８に示したように、時刻ｔ１以前においては、ＯＳにＣＰＵ１１の占有権があり、タスクＡ～Ｃのすべてが待ち状態である。いずれかのタスクがレディ状態になるのをＯＳが待っている状態である。この状態では、図９に示したように、ＣＰＵ１１のプログラムカウンタはＯＳ自身のルーチンのうちのいずれかのアドレスを指しており、ＯＳ用スタック領域ＳＯが使用されている。スタックポインタＳＰは、ＯＳ用スタック領域ＳＯの使用中領域のうちのいずれかのアドレスを指している。このとき、Ａ用ＳＰバッファには、Ａ用スタック領域ＳＡのスタートアドレスがセットされ、Ｂ用ＳＰバッファおよびＣ用ＳＰバッファには、共にＢＣ共用スタック領域ＳＢＣのスタートアドレスがセットされている。

【００３８】ここで、時刻ｔ１においてＯＳ基準割込信号による割込みがあると、タスクＣは直ちにレディ状態となる。ＯＳはＣＰＵ１１の占有権をどのタスクに渡すべきかを判断する。この時点では、優先順位の最も高いタスクＡおよび次に優先順位の高いタスクＢは共に待ち状態になっているため、図１０に示したように、ＯＳはスタックポインタＳＰの示すアドレスをＢＣ共用スタック領域ＳＢＣのスタートアドレスにセットすると共に、プログラムカウンタをタスクＣの最初のアドレスにセットし、ＣＰＵ１１の占有権をタスクＣに渡す。これにより、タスクＣは実行状態となる。このとき、Ａ用ＳＰバッファには、Ａ用スタック領域ＳＡのスタートアドレスがセットされ、Ｂ用ＳＰバッファおよびＣ用ＳＰバッファには、共にＢＣ共用スタック領域ＳＢＣのスタートアドレスがセットされている。

【００３９】時刻ｔ１でタスクＣが実行状態となったのち、このタスクＣよりも優先順位の高いタスクＢについて設定されていた起動待ちタイマがタイムアップしたとする。従来の方法では、ここでタスクＢがタスクＣを中断して実行状態となっていたが、本実施の形態では、図８（ｃ）に示したように、タスクＢはタスクＣを中断することなく、単に、待ち状態からレディ状態に遷移するのみである。

【００４０】さらに、時間が経過して、時刻ｔ２において垂直プランキングパルス４２による割込みがかかると、ＯＳは、図１１に示したように、この時点でのプログラムカウンタ等のＣＰＵ１１のレジスタ群の内容をＢＣ共用スタック領域ＳＢＣに退避（プッシュ）すると共に、このときのスタックポインタＳＰの値をＢ用ＳＰバッファおよびＣ用ＳＰバッファに格納する。これにより、Ｂ用ＳＰバッファおよびＣ用ＳＰバッファの内容は、ＢＣ共用スタック領域ＳＢＣのうちの使用されている領域の最終アドレスとなる。そして、ＯＳは、Ａ用ＳＰバッファに格納されているスタックポインタＳＰの値をＡ用スタック領域ＳＡのスタートアドレスとし、さらに、プログラムカウンタをタスクＡの先頭アドレスにセットする。これにより、タスクＡはレディ状態を経て実行状態に遷移し、タスクＣは実行状態からＳＰＬレディ状態に遷移する。なお、タスクＢはそのままレディ状態を維持する。

【００４１】タスクＡは、時刻ｔ３において必要な処理が終了すると、自らを待ち状態とするためのウェイト命

令をＯＳに出し、ＯＳにＣＰＵ１１を開放して自らは待ち状態となる。このときのタスクＢの状態はレディ状態であり、タスクＣの状態はＳＰＬレディ状態となっている。このため、ＯＳは、より優先順位の高いタスクＢを差し置いてタスクＣを優先して実行状態に遷移させ、タスクＡにより中断されていた処理を再開させる。具体的には、ＯＳは、ＢＣ共用スタック領域ＳＢＣに退避されていた各種のレジスタデータをＣＰＵ１１のレジスタ群に復帰（ポップ）させる。これにより、図１２に示したように、プログラムカウンタはタスクＣの再開アドレスにセットされる。また、このとき、Ｂ用ＳＰバッファおよびＣ用ＳＰバッファに格納されているアドレス（この場合、タスクＣがタスクＡによって実行を中断された時点の最終アドレス）をスタックポインタＳＰに復帰させ、各バッファはＢＣ共用スタック領域ＳＢＣの先頭アドレスに戻される。この点が本発明の特徴の１つをなしている。なお、このときのＡ用ＳＰバッファの内容はＡ用スタック領域ＳＡ のスタートアドレスとなり、Ｂ用ＳＰバッファおよびＣ用ＳＰバッファの内容は、ＢＣ共用スタック領域ＳＢＣのうちの使用されている領域の先頭アドレスとなり、また、スタックポインタＳＰはＢＣ共用スタック領域ＳＢＣ内におけるタスクＣが使用した最終アドレスとなる。

【００４２】実行を再開したタスクＣは、時刻ｔ４において必要な処理が終了すると、自らを待ち状態にすることを求めるウェイト命令をＯＳに出してＣＰＵ１１を開放する。すると、ＯＳは、この時点で唯一レディ状態となっているタスクＢを実行状態に遷移させる。具体的には、図１３に示したように、プログラムカウンタをタスクＢの先頭アドレスにセットすると共に、上記したようにＢＣ共用スタック領域ＳＢＣの先頭アドレスに戻されたＢ用ＳＰバッファおよびＣ用ＳＰバッファの値を、スタックポインタＳＰにコピーする。これにより、これ以降、ＢＣ共用スタック領域ＳＢＣはタスクＢによって使用可能となる。なお、このとき、タスクＡは待ち状態を維持し、Ａ用ＳＰバッファの内容はＡ用スタック領域ＳＡ のスタートアドレスとなっている。

【００４３】実行状態となったタスクＢは、時刻ｔ５において必要な処理を終了させると、自らを待ち状態にすることを求めるウェイト命令をＯＳに出してＣＰＵ１１をＯＳに開放する。これにより、状況１における処理が終わる。

【００４４】このように、あるタスクが実行状態である場合には、属性１以外の属性をもつより優先順位の高いタスクから割込要求がなされたとしても、実行状態のタスクはこのタスクにＣＰＵ１１の占有権を渡すことはなく、実行途中のタスクが終了するまで、その優先順位の高いタスクは待つこととなる。但し、属性１のタスクからの割込要求についてはこの限りでなく、要求に応じて直ちにＣＰＵ１１の占有権が与えられることとなる。また、レディ状態にあるタスクとＳＰＬレディ状態にあるタスクとが競合したときは、優先順位の上下に係わりなく、ＳＰＬレディ状態にあるタスクにＣＰＵ１１の占有権が与えられることとなる。

【００４５】次に、図１４を参照して、図７における状況２を説明する。

【００４６】ここでは、時刻ｔ６において、ＯＳ基準割込信号による割込みが行われると同時に、タスクＢ用の起動待ちタイマがタイムアップしたとする。この場合、タスクＢおよびタスクＣの双方がレディ状態となっているが、タスクＣよりもタスクＢの方が優先順位が高く設定されている。このため、タスクＢが実行状態となってＣＰＵ１１を占有する。一方、タスクＣはレディ状態となって、タスクＢが実行を終了してＣＰＵ１１を開放するのを待つ。時刻ｔ７においてタスクＢによる必要な処理が終了すると、タスクＢは休眠状態になる一方、タスクＣは実行状態になる。そして、タスクＣは、必要な処理を行ったのち、時刻ｔ８においてＣＰＵ１１をＯＳに開放し、実行状態から待ち状態に遷移する。

【００４７】このように、タスクＢとタスクＣとが共にレディ状態にあるときに、起動条件が競合したときは、原則通り、優先順位に従って、より高い優先順位のタスクに対してＣＰＵ１１の占有権が与えられることとなる。

【００４８】以上のように、本実施の形態に係るマルチタスク制御方法では、タスクＢとタスクＣとが共にレディ状態にあって起動要求が競合したときは、原則通り、優先順位に従って、より高い優先順位のタスクに対してＣＰＵ１１の占有権が与えられるが、属性２のタスクＢまたはＣの実行中に属性１のタスクＡから割込要求があったときは、直ちにＣＰＵ１１の占有権をタスクＡに与える一方、タスクＢ（またはＣ）の実行中にタスクＣ（またはＢ）からの割込み要求があったときは、そのタスクの優先順位の如何に係わらず、その実行中のタスクＢ（またはＣ）の終了を待ってＣＰＵ１１の占有権を開放するようにしたので、必要なタスク処理のみを限定的にリアルタイム動作させ、さほどリアルタイム性を必要としないタスクについてはいわば擬似的なリアルタイム動作をさせることができる。このため、不要な機能についての処理プログラム部分を削減することができ、ＯＳ自体のプログラム構成をシンプルにできる。また、属性１以外のタスク（ＢまたはＣ）については、同時に同一のスタック領域を使用することがなくなるので、スタック領域を複数のタスク（ＢおよびＣ）によって共用することが可能となる。

【００４９】また、レディ状態にあるタスクとＳＰＬレディ状態にあるタスクとが競合したときは、優先順位の上下に係わりなく、必ず、ＳＰＬレディ状態にあるタスクにＣＰＵ１１の占有権が与えられるようにしたので、タスクＡによって実行を中断されたタスクＢまたはＣ

は、タスクAの実行終了後、確実に実行を再開することができる。

【0050】なお、以上のような動作を行うOSは、例えば次のようにして所定の記録媒体に記録可能である。すなわち、まず、例えばC言語を用いてOS本体部および各タスクに係るプログラムを作成する。次に、これらのプログラムをコンパイラによりそれぞれコンパイルして、アセンブリ言語で記述されたプログラムに変換する。次に、これらのアセンブリ言語で記述されたプログラムをアセンブラを用いてそれぞれマシン語に変換すると共に、これらの複数のマシン語プログラムをリンカを用いて統合して1つのプログラムとする。こうして作成されたOS全体のプログラムは、例えばフロッピーディスク、マスクROM、EPROM（消去・再書込可能ROM）、またはOTP（ワンタイムPROM）等に書き込まれ、あるいは、マイクロコンピュータのチップ自体に組み込まれて、一般に提供可能となる。

【0051】以上、いくつかの実施の形態を挙げて本発明を説明したが、本発明はこれらの実施の形態に限定されず、種々変形可能である。例えば、上記実施の形態では、属性2のタスクはタスクBおよびタスクCの2つとしたが、それ以上の数のタスクに対しても適用可能である。また上記の実施の形態では、タスクAは垂直ブランキングパルス42による割込みによって起動することとしたが、他の割込み信号によって起動されるようにしてもよい。また、本実施の形態では、電子機器として映像表示装置を例にとって説明したが、他の電子機器にも同様に適用可能である。

【0052】

【発明の効果】以上説明したように本発明に係るマルチタスク制御方法によれば、即時実行が要求される特定のタスクに対してのみ限定的にリアルタイム制御を行い、それ以外のタスクに対しては擬似的なリアルリイム制御を行うようにしたので、すべてのタスクに対して平等にリアルタイム制御を行うようにした従来の汎用マルチタスク制御方法に比べて、それを実現するためのプログラムのサイズが小さくて済み、プログラム格納用のメモリ容量を削減することができる。しかも、上記した特定のタスク以外のタスクについてはスタック領域を共用化することができるので、スタックメモリの容量をも削減することができる。したがって、マルチタスク制御に必要なメモリ容量を削減することができる。さらに、その特定のタスク以外のタスクに関してはリアルタイム性をさほど要求しないシステムにとっては、メモリ資源が無駄に消費されるのを防止することができる。また、本発明のマルチタスク制御方法によれば、その機能構成が単純であるので、それを実現するためのプログラムの殆どを例えばC言語等によって開発することができ、これにより、異なるマイクロコンピュータシステムへの移植が容易となる。

【図面の簡単な説明】

【図1】本発明の一実施の形態に係るマルチタスク制御方法が適用される電子機器としての映像表示装置の概略構成を表すブロック図である。

【図2】図1の映像表示装置における主制御部の概略構成を表すブロック図である。

【図3】図2のCPUの動作を制御するオペレーティングシステムと各タスクとの関係を表す図である。

【図4】図2のCPU内のレジスタ群とROMおよびRAM内のメモリアドレスとの関係を表す図である。

【図5】タスクの状態遷移の様子を表す図である。

【図6】タスクの状態遷移を説明するための流れ図である。

【図7】各タスクの動作タイミングの一例を表すタイミング図である。

【図8】図7における一部の動作タイミングを詳細に表す図である。

【図9】図8に示した動作を行っているCPUおよびRAMのある時点での状態を表す図である。

【図10】図8に示した動作を行っているCPUおよびRAMの他の状態を表す図である。

【図11】図8に示した動作を行っているCPUおよびRAMのさらに他の状態を表す図である。

【図12】図8に示した動作を行っているCPUおよびRAMのさらに他の状態を表す図である。

【図13】図8に示した動作を行っているCPUおよびRAMのさらに他の状態を表す図である。

【図14】図7における他の部分の動作タイミングを詳細に表す図である。

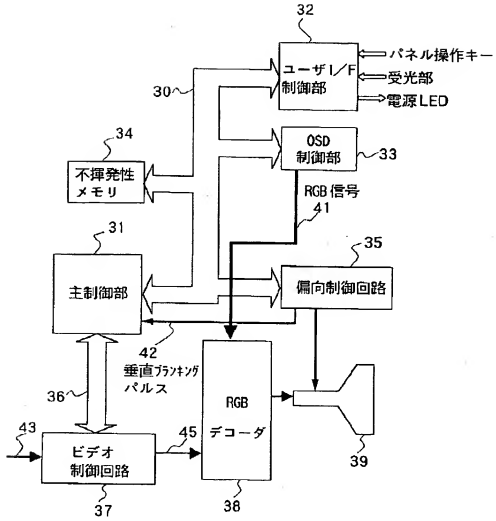
【図15】従来のマルチタスク制御方法の一例を説明するためのタイミング図である。

【図16】従来のマルチタスク制御方法におけるスタック領域の構成例を表す図である。

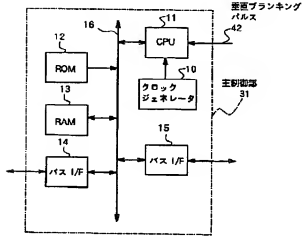
【符号の説明】

1，2…OS基準割込信号、11…CPU、12…ROM、13…RAM、31…主制御部、42…垂直ブランキングパルス、SA…A用スタック領域、SBC…BC共用スタック領域、S0…OS用スタック領域、A～C…タスク

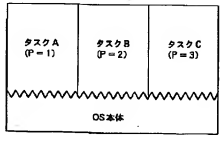
【図１】


32
ユーザI/F
制御部
パネル操作キー
受光部
電源LED
30
34
不揮発性
メモリ
OSD
制御部
33
RGB信号
41
31
主制御部
35
偏向制御回路
42
垂直ブランキング
パルス
36
43
ビデオ
制御回路
45
RGB
デコーダ
39
37
38

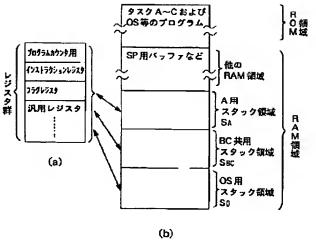
【図２】


垂直ブランキング
パルス
16
11
CPU
42
12
ROM
10
クロック
ジェネレータ
主制御部
31
13
RAM
15
14
バス I/F
バス I/F

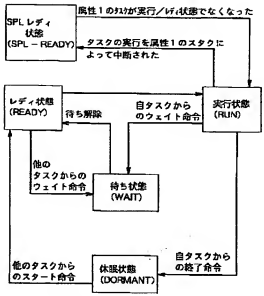
【図３】


タスクA
(P＝1)
タスクB
(P＝2)
タスクC
(P＝3)
OS本体

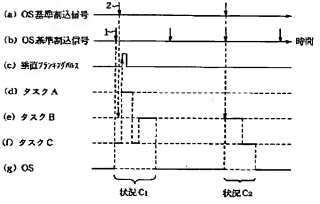
【図４】



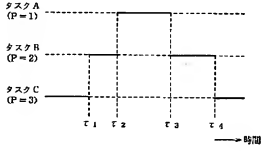
【図５】



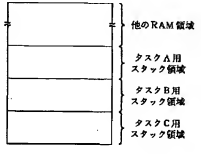
【図７】



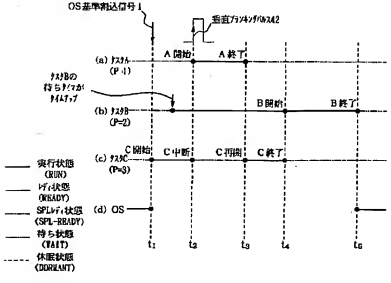
【図１５】



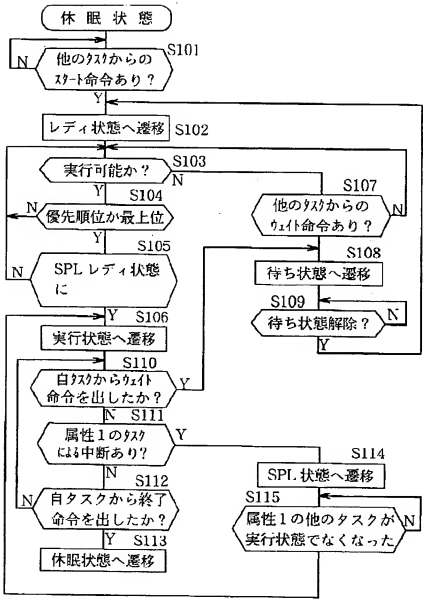
【図１６】



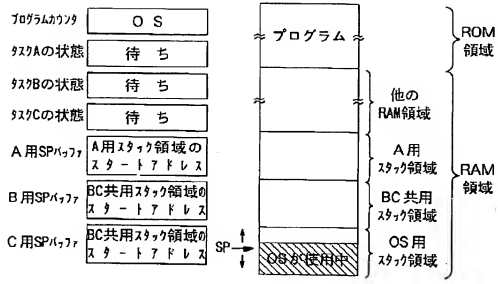
【図８】



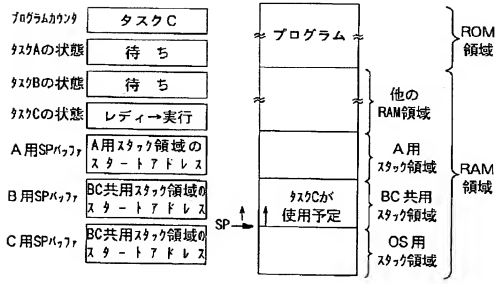
【図６】


休眠状態
S101
他のﾀｽｸからのｽﾀｰﾄ命令あり？
N
Y
レディ状態へ遷移
S102
実行可能か？
S103
N
Y
S104
優先順位が最上位
N
Y
S105
SPL レディ状態に
N
Y
S106
実行状態へ遷移
S107
他のﾀｽｸからのｳｪｲﾄ命令あり？
N
S108
待ち状態へ遷移
S109
待ち状態解除？
N
Y
S110
自ﾀｽｸからｳｪｲﾄ命令を出したか？
Y
N
S111
属性１のﾀｽｸによる中断あり？
Y
N
S112
自タスクから終了命令を出したか？
N
Y
S113
休眠状態へ遷移
S114
SPI. 状態へ遷移
S115
属性１の他のタスクが実行状態でなくなった
N

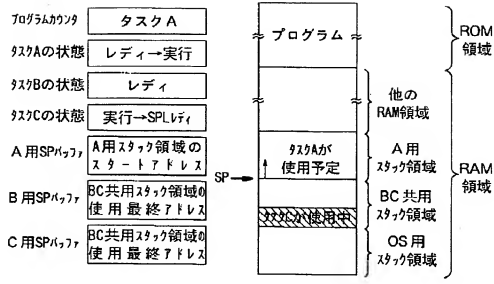
【図9】


プログラムカウンタ
OS
タスクAの状態
待ち
タスクBの状態
待ち
タスクCの状態
待ち
A用SPバッファ
A用スタック領域のスタートアドレス
B用SPバッファ
BC共用スタック領域のスタートアドレス
C用SPバッファ
BC共用スタック領域のスタートアドレス
SP
≈プログラム≈
OSが使用中
他のRAM領域
A用スタック領域
BC共用スタック領域
OS用スタック領域
ROM領域
RAM領域

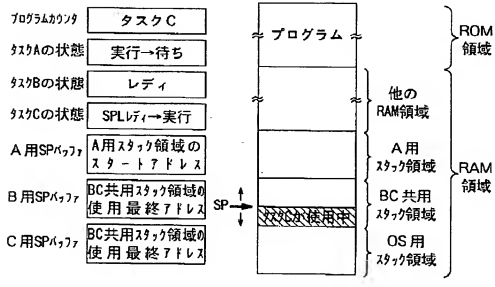
【図10】


プログラムカウンタ
タスクC
タスクAの状態
待ち
タスクBの状態
待ち
タスクCの状態
レディ→実行
A用SPバッファ
A用スタック領域のスタートアドレス
B用SPバッファ
BC共用スタック領域のスタートアドレス
C用SPバッファ
BC共用スタック領域のスタートアドレス
SP
≈プログラム≈
タスクCが使用予定
他のRAM領域
A用スタック領域
BC共用スタック領域
OS用スタック領域
ROM領域
RAM領域

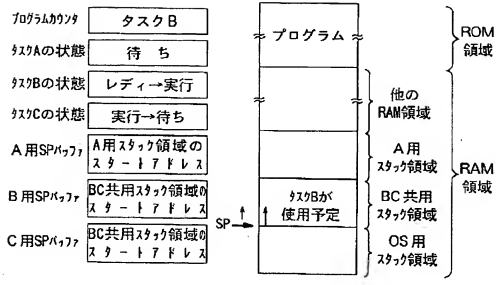
【図１１】


プログラムカウンタ
タスクA
タスクAの状態
レディ→実行
タスクBの状態
レディ
タスクCの状態
実行→SPLレディ
A用SPバッファ
A用スタック領域のスタートアドレス
B用SPバッファ
BC共用スタック領域の使用最終アドレス
C用SPバッファ
BC共用スタック領域の使用最終アドレス
SP
プログラム
タスクAが使用予定
タスクCが使用中
他のRAM領域
A用スタック領域
BC共用スタック領域
OS用スタック領域
ROM領域
RAM領域

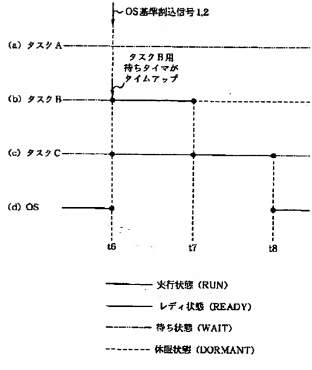
【図１２】


プログラムカウンタ
タスクC
タスクAの状態
実行→待ち
タスクBの状態
レディ
タスクCの状態
SPLレディ→実行
A用SPバッファ
A用スタック領域のスタートアドレス
B用SPバッファ
BC共用スタック領域の使用最終アドレス
C用SPバッファ
BC共用スタック領域の使用最終アドレス
SP
プログラム
タスクCが使用中
他のRAM領域
A用スタック領域
BC共用スタック領域
OS用スタック領域
ROM領域
RAM領域

【図１３】


プログラムカウンタ
タスクAの状態
タスクBの状態
タスクCの状態
A用SPバッファ
B用SPバッファ
C用SPバッファ
タスクB
待ち
レディ→実行
実行→待ち
A用スタック領域のスタートアドレス
BC共用スタック領域のスタートアドレス
BC共用スタック領域のスタートアドレス
SP
プログラム
タスクBが使用予定
他のRAM領域
A用スタック領域
BC共用スタック領域
OS用スタック領域
ROM領域
RAM領域

【図１４】


OS基準割込信号1.2
(a) タスクA
タスクB用待ちタイマがタイムアップ
(b) タスクB
(c) タスクC
(d) OS
t6
t7
t8
実行状態 (RUN)
レディ状態 (READY)
待ち状態 (WAIT)
休眠状態 (DORMANT)